Available online at www.sciencedirect.com
ScienceDirect
Discrete Optimization 5 (2008) 530–540
DISCRETE OPTIMIZATION
www.elsevier.com/locate/disopt

An improved algorithm for computing Steiner minimal trees in Euclidean d-space

Marcia Fampa[a], Kurt M. Anstreicher[b,*]

[a] *Institute of Mathematics, Department of Computer Sciences, Federal University of Rio de Janeiro, CP 68.530, 21941-590 Rio de Janeiro, RJ, Brazil*
[b] *Department of Management Sciences, University of Iowa, Iowa City, IA 52242, USA*

Received 17 April 2006; accepted 27 August 2007
Available online 25 October 2007

Abstract

We describe improvements to Smith's branch-and-bound (B&B) algorithm for the Euclidean Steiner problem in $\mathbb{R}^d$. Nodes in the B&B tree correspond to full Steiner topologies associated with a subset of the terminal nodes, and branching is accomplished by "merging" a new terminal node with each edge in the current Steiner tree. For a given topology we use a conic formulation for the problem of locating the Steiner points to obtain a rigorous lower bound on the minimal tree length. We also show how to obtain lower bounds on the child problems at a given node without actually computing the minimal Steiner trees associated with the child topologies. These lower bounds reduce the number of children created and also permit the implementation of a "strong branching" strategy that varies the order in which the terminal nodes are added. Computational results demonstrate substantial gains compared to Smith's original algorithm.

Keywords: Euclidean Steiner problem; Steiner tree; Branch and bound; Strong branching

1. Introduction

The Euclidean Steiner tree problem (ESTP) in $\mathbb{R}^d$ is defined as follows: Given a set of points in $\mathbb{R}^d$, find a tree of minimal Euclidean length that spans these points but that can also utilize additional points in the construction of the tree. We refer to the original points as *terminal nodes*, and any additional nodes in the spanning tree as *Steiner points*. The ESTP has been shown to be NP-Hard [9] and has received considerable attention in the literature. For a comprehensive survey see [12]. The solution of an ESTP is called a Steiner minimal tree (SMT). Some well-known basic properties of SMTs are:

- A Steiner point in an SMT has degree equal to three. The Steiner point and its 3 adjacent nodes lie in a plane, and the included angles between the arcs connecting the point to its adjacent nodes are all 120 degrees.
- A terminal node in an SMT has degree between one and three.
- An SMT for a problem with N terminal nodes has at most $N - 2$ Steiner points.

* Corresponding author. Tel.: +1 319 335 0890; fax: +1 319 335 0297.
E-mail addresses: fampa@cos.ufrj.br (M. Fampa), kurt-anstreicher@uiowa.edu (K.M. Anstreicher).

1572-5286/$ - see front matter
doi:10.1016/j.disopt.2007.08.006

We define the *topology* of a Steiner tree to be the tree for which we have fixed the number of Steiner points and the edges between the points, but not the geometric position of the Steiner points. A topology is called a *Steiner topology* if each Steiner point has degree equal to three and each terminal node has degree three or less. A Steiner topology with N terminal nodes is a *full Steiner topology* if there are $N-2$ Steiner points and each terminal node has degree equal to one. A Steiner tree which corresponds to some topology, but with certain edges shrunk to zero length, is said to be degenerate. Any SMT with a nonfull Steiner topology can also be associated with a full Steiner topology for which the tree is degenerate.

A number of papers have considered the exact solution of the ESTP in $\mathbb{R}^2$ [5,6,11–13,18,22–24]. Melzak [18] was the first to present an algorithm to solve the problem, which was based on the enumeration of all Steiner topologies and on the determination of the length of the Steiner minimal tree corresponding to each topology. At the present time the best exact solution algorithm for the ESTP in the plane, the GeoSteiner algorithm created by Warme, Winter and Zachariasen [22], has the ability to handle typical problem instances with thousands of terminals. Methods specialized for the ESTP in $\mathbb{R}^2$ cannot be applied to problems in higher dimensions, however, and very few papers have considered exact methods for $d \geq 3$. A polynomial time approximation scheme (PTAS) is known for the ESTP, see [3]. The running time of a randomized version of this PTAS to obtain a $(1+1/c)$-approximation of an SMT for a problem with N terminal nodes in $\mathbb{R}^d$ is of the form $O(N(\log N)^{O((\sqrt{d}c)^{d-1})})$.

Gilbert and Pollak [10] proposed computing SMTs in $\mathbb{R}^d$ by enumerating all Steiner topologies and computing the minimal length for the tree associated with each topology. Unfortunately the number of Steiner topologies having N terminal nodes grows extremely fast in N, so the enumeration of all topologies is only possible for very small values of N. Maculan et al. [17] formulated the ESTP as a nonconvex mixed-integer program and proposed a Branch and Bound (B&B) algorithm using Lagrangian dual bounds. Fampa and Maculan [8] presented a convex mixed-integer formulation that could be implemented in a B&B algorithm using bounds computable from conic problems. The formulations in [17,8] both use binary variables to indicate whether or not the edge connecting two nodes is present in a Steiner topology. The presence of these binary variables leads to a natural branching scheme, but it can be expected that the bounds obtained when integrality is relaxed will be very weak. Neither [17] nor [8] present computational results based on the proposed formulations.

A different B&B algorithm for finding Steiner minimal trees in $\mathbb{R}^d$ is described by Smith [19]. The key to Smith's algorithm is a novel implicit enumeration scheme for all full Steiner topologies on a given set of N terminals. The root node in the B&B tree corresponds to the unique full Steiner topology for three given terminals, and the nodes at depth k in the B&B tree enumerate all full Steiner topologies having $k+3$ terminal nodes. Branching is accomplished by adding a new terminal node and creating children whose topologies are obtained by "merging" a given edge in the tree with the new terminal node to create a new full Steiner topology. It is easily shown that the merging operation cannot *decrease* the minimal length of the tree, and therefore if the minimal length of a Steiner tree at some node in the B&B tree is *longer* than that of a known Steiner tree on all N terminals the given node may be "fathomed" and its descendants removed from further consideration. If a node cannot be fathomed, a new terminal is added to the given topology and one child node is created for each one of its edges. Computational results in [19] obtain SMTs, or good lower bounds on their lengths, for the vertices of regular d-polytopes with 16 or fewer vertices. Results for the simplex and octahedron in dimensions $3 \leq d \leq 9$ are sufficient to disprove a conjecture of Gilbert and Pollak on the "Steiner ratio" in these dimensions. (The Steiner ratio in $\mathbb{R}^d$ is the minimal possible ratio between the length of an SMT for a given set of terminals in $\mathbb{R}^d$ and the length of a minimal spanning tree on the same terminals which is not permitted to add any additional points.) The results of [19] strongly suggested that the Gilbert–Pollak conjecture was false for all $d \geq 3$, which was subsequently proved by Du and Smith [7]. Additional computations [21] support a new conjecture on the Steiner ratio in $\mathbb{R}^3$.

The algorithm of [19] represents the current state of the art for computing SMTs in $\mathbb{R}^d$, $d \geq 3$. Nevertheless, the method has several deficiencies. For example, the problem that must be solved to generate a bound at each node is convex but possibly nondifferentiable, creating difficulties for many nonlinear programming methods. The "bounds" used by Smith do not in fact correspond to rigorous lower bounds on the solution values of these problems, but are instead obtained from putatively near-optimal solutions. In addition, when a node fails to fathom the algorithm has no means to estimate the objective values associated with its potential children. As a result the terminal nodes are added in a fixed order, even though varying the order has the potential to substantially reduce the size of the B&B tree. In this paper we consider enhancements to Smith's algorithm that address these issues. Our improved method

uses a conic formulation of the problem associated with each B&B node and obtains a rigorous lower bound obtained via a dual solution. We also introduce a methodology for estimating the effect of merging a new terminal node to a given arc that requires substantially less computation than actually solving the optimization problem associated with the new augmented topology. The availability of these bounds allows us to eliminate some children and also provides the information necessary to implement a "strong branching" scheme that varies the order in which the terminal nodes are added in an effort to reduce the size of the B&B tree. Computational results demonstrate substantial improvements over Smith's original algorithm.

Notation. For matrices A and B, $A \otimes B$ denotes the Kronecker product, $\mathbf{vec}(A)$ denotes the vector obtained by "stacking" the columns of A in the natural order, and (for A and B having the same dimensions) $A \bullet B$ denotes the matrix inner product $\mathbf{vec}(A)^{\mathrm{T}}\mathbf{vec}(B)$. For any optimization problem (Q), $z^*(Q)$ denotes the optimal solution value of Q.

2. A new lower bound for descendants of a partial Steiner topology

Let T be a tree with full Steiner topology in $\mathbb{R}^d$ having m Steiner points, $m+2$ terminal nodes, and $n = 2m+1$ edges. When the terminal nodes in T are a subset of the $N > m+2$ terminal nodes in a larger problem we call T a *partial Steiner tree*, and its topology a *partial Steiner topology*. Let Y be a $d \times m$ matrix whose ith column y_i is the location of the ith Steiner point. Let A and C be $m \times n$ and $d \times n$ matrices, respectively, whose ith columns a_i and c_i are defined as follows. If edge i is adjacent to one terminal node and Steiner point j, then c_i contains the coordinates of the adjacent terminal node and $a_i = e_j$, where $e_j \in \mathbb{R}^m$ is the jth elementary vector. If edge i is adjacent to two Steiner points j and k with $j < k$, then $c_i = 0$ and $a_i = e_k - e_j$. For the given full Steiner topology, the problem of determining the locations of the Steiner points can then be written

$$\begin{array}{lll} (P) & \text{Min} & \displaystyle\sum_{i=1}^{n} \|s_i\| \\ & \text{s.t.} & YA + S = C, \end{array}$$

where S is a $d \times n$ matrix whose ith column s_i is the difference in the locations of the two nodes adjacent to edge i in the tree. To obtain the dual of (P) it is convenient to re-write the problem using a vector $y = \mathbf{vec}(Y) \in \mathbb{R}^{dm}$ for the locations of the Steiner points. In this form the problem of locating the Steiner points can be written

$$\begin{array}{lll} (P) & \text{Min} & \displaystyle\sum_{i=1}^{n} \|s_i\| \\ & \text{s.t.} & A_i^{\mathrm{T}} y + s_i = c_i, \quad i = 1, \ldots, n, \end{array}$$

where $A_i^{\mathrm{T}} = a_i^{\mathrm{T}} \otimes I_d$ and I_d is a $d \times d$ identity matrix. Problem (P) is a "minimum sum of norms" problem and can also be viewed as a Euclidean multifacility location problem (EMLP). The dual of (P) is given by [2]

$$\begin{array}{lll} (D) & \text{Max} & \displaystyle\sum_{i=1}^{n} c_i^{\mathrm{T}} x_i \\ & \text{s.t.} & \displaystyle\sum_{i=1}^{n} A_i x_i = 0, \\ & & \|x_i\| \le 1, \quad i = 1, \ldots, n. \end{array}$$

Letting X be a $d \times n$ matrix whose ith column is x_i, problem (D) can also be written

$$\begin{array}{lll} (D) & \text{Max} & C \bullet X \\ & \text{s.t.} & XA^{\mathrm{T}} = 0 \\ & & \|x_i\| \le 1, \quad i = 1, \ldots, n. \end{array}$$

If one regards each component of x_i as a flow on edge i, then the equality constraints of (D) have an interpretation as d-dimensional flow balance constraints at each of the Steiner nodes. It is known [2] that if $(\bar{Y}, \bar{S})$ is an *optimal* solution to (P) with $\bar{s}_i \neq 0$, then $\bar{x}_i = \bar{s}_i / \|\bar{s}_i\|$ in an optimal solution to (D).

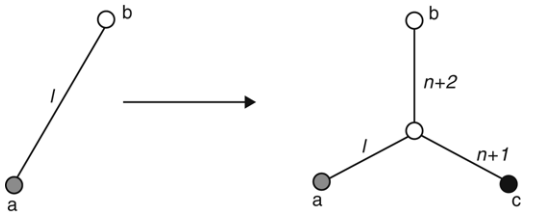

Fig. 1. Generating tree T^+ from T.

In the sequel, we will be interested in solving restrictions of (D) where the variables x_i are fixed at values $\bar{x}_i$ for $i \in \bar{E} \subset \{1, \ldots, n\}$. Let $E = \{1, \ldots, n\} \setminus \bar{E}$. The resulting problem can be written

$$
\begin{array}{lll}
(\bar{D}) & \text{Max} & \bar{v} + \sum_{i \in E} c_i^{\mathrm{T}} x_i \\
& \text{s.t.} & \sum_{i \in E} A_i x_i = \bar{b}, \\
& & \|x_i\| \leq 1, \quad i \in E,
\end{array}
$$

where $\bar{b} = -\sum_{i \in \bar{E}} A_i \bar{x}_i$ and $\bar{v} = \sum_{i \in \bar{E}} c_i^{\mathrm{T}} \bar{x}_i$. The dual of $(\bar{D})$ can easily be obtained by using a conic formulation of the problem. To do this we first embed each x_i in a $(d+1)$-dimensional vector $(\tau_i, x_i^{\mathrm{T}})^{\mathrm{T}}$ which lies in the second-order cone $\|x_i\| \leq \tau_i$, and then add the linear constraints $\tau_i = 1, i \in E$. The dual of the resulting second-order cone program (SOCP) is easily shown to be [15]

$$
\begin{array}{lll}
(\bar{P}) & \text{Min} & \bar{v} + \sum_{i \in E} \|s_i\| + \bar{b}^{\mathrm{T}} y \\
& \text{s.t.} & A_i^{\mathrm{T}} y + s_i = c_i, \quad i \in E.
\end{array}
$$

Note that $(\bar{P})$ includes the locations of all of the Steiner points as variables. However, it is easy to see that if $\bar{E}$ includes all three of the arcs incident to some Steiner node i, then all coefficients of y_i in the objective and constraints of $(\bar{P})$ are zero, and therefore y_i can be removed from the problem. It is also obvious that if $\bar{E}$ includes two of the arcs incident to some Steiner node i then the value of x_k, where k is the index of the third arc incident to node i, is forced by the equality constraints of $(\bar{D})$. As a result we may assume w.l.o.g. that $\bar{E}$ contains either zero, one, or all three of the arcs incident to each Steiner node.

Now suppose that a tree T^+ is generated from tree T by replacing an edge l, adjacent to two nodes a and b, with a new Steiner node $m+1$ that is adjacent to a, b, and a new terminal node. This procedure is illustrated in Fig. 1 and will be referred to as "merging" edge l with the new terminal node.[1] (We assume throughout, without loss of generality, that b is a Steiner node. In figures we use an open circle to denote a Steiner node, a black circle to denote a terminal node, and a gray circle to denote a node that could be either a terminal or a Steiner node.) We use (P^+) and (D^+) to denote the problems (P) and (D) associated with the augmented tree T^+. In particular we can write (D^+) as

$$
\begin{array}{lll}
(D^+) & \text{Max} & C^+ \bullet X^+ \\
& \text{s.t.} & X^+ A^{+T} = 0 \\
& & \|x_i\| \leq 1, \quad i = 1, \ldots, n+2,
\end{array}
$$

where X^+ is the $d \times (n+2)$ matrix whose jth column is x_j. The data (A^+, C^+) is related to the original data (A, C) as follows. We distinguish two cases, depending on whether or not the edge l is adjacent to a terminal node. Clearly $a_i^+ = (a_i^{\mathrm{T}}, 0)^{\mathrm{T}}$ and $c_i^+ = c_i$ for $1 \leq i \neq l \leq n$.

Case 1: Edge l is adjacent to two Steiner nodes. In this case let a and b denote Steiner nodes j and k, respectively, with $j < k$. Then c_{n+1}^+ contains the coordinates of the new terminal node, $c_l^+ = c_{n+2}^+ = 0$, and

[1] Smith [19] calls this the "sprout" operation.

$$a_l^+ = e_{m+1} - e_j,$$
$$a_{n+1}^+ = e_{m+1},$$
$$a_{n+2}^+ = e_{m+1} - e_k.$$

Case 2: Edge l is adjacent to a terminal node. In this case let a be the terminal node and let b denote Steiner node j. Then c_{n+1}^+ contains the coordinates of the new terminal node, $c_l^+ = c_l$, $c_{n+2}^+ = 0$, and

$$a_l^+ = e_{m+1},$$
$$a_{n+1}^+ = e_{m+1},$$
$$a_{n+2}^+ = e_{m+1} - e_j.$$

Let $\bar{x}$ be an optimal solution to (D). It is well-known (and will be proved below) that $z^*(P^+) \geq z^*(P)$. Our goal here is to use $\bar{x}$ to obtain a valid lower bound on $z^*(P^+)$ that is at least as good as $z^*(P)$ but that requires less work to compute than solving (P^+). To do this we consider the problem (D^+) with fixed values $x_i = \bar{x}_i$, $i \in \bar{E}$, where $l \notin \bar{E}$. Let $E^+ = \{1, \ldots, n+2\} \setminus \bar{E}$. The resulting problem $(\bar{D}^+)$ can be written

$$\begin{array}{lll} (\bar{D}^+) & \text{Max} & \bar{v} + \sum_{i \in E^+} c_i^{+T} x_i \\ & \text{s.t.} & \sum_{i \in E^+} A_i^+ x_i = \bar{b}^+ \\ & & \|x_i\| \leq 1, \quad i \in E^+, \end{array}$$

where $A_i^+ = a_i^+ \otimes I_d$ for each i, and $\bar{b}^+ = (\bar{b}^{\mathrm{T}}, 0)^{\mathrm{T}} = -\sum_{i \in \bar{E}} A_i^+ \bar{x}_i$.

Theorem 1. *Let $\bar{x}$ be an optimal solution of (D). Then $z^*(P) = z^*(\bar{P}) \leq z^*(\bar{P}^+) \leq z^*(P^+)$.*

Proof. Since $\bar{x}$ is an optimal solution to (D) it is obvious that $z^*(D) = z^*(\bar{D})$, so $\bar{x}_i$, $i \in E$ is also optimal in $(\bar{D})$ and $z^*(P) = z^*(\bar{P})$. It is also obvious that $z^*(\bar{D}^+) \leq z^*(D^+)$ since $(\bar{D}^+)$ is a restriction of (D^+), and therefore $z^*(\bar{P}^+) \leq z^*(P^+)$. It remains to show that $z^*(\bar{P}) \leq z^*(\bar{P}^+)$. To do this we will show that $z^*(\bar{D}) \leq z^*(\bar{D}^+)$ by constructing a feasible solution x_i, $i \in E^+$ for $(\bar{D}^+)$ whose objective value is also equal to $z^*(D)$. Let $x_i = \bar{x}_i$, $i \in E$, $x_{n+1} = 0$, $x_{n+2} = -\bar{x}_l$. It is then straightforward to verify that in both Case 1 and Case 2 the solution x_i, $i \in E^+$ is feasible in $(\bar{D}^+)$, and $\bar{v} + \sum_{i \in E^+} c_i^{+T} x_i = \sum_{i=1}^n c_i^T \bar{x}_i = z^*(D)$. □

Note that for a given partial topology and a new terminal node, every descendant of this topology using Smith's enumeration scheme corresponds to merging the new terminal node with one of the n arcs in the current tree. The following corollary uses this fact together with Theorem 1 to obtain an improved lower bound for the minimal total length of any Steiner tree that is a descendant of the current partial topology.

Corollary 2. *Let z^{**} be the minimal length of a Steiner tree that is a descendant of the partial tree T associated with problem (P), using Smith's enumeration scheme. Let $\bar{x}_i$, $i = 1, \ldots, n$ be an optimal solution to (D). For $1 \leq l \leq n$, denote by (D_l^+) the problem corresponding to a tree T_l^+ generated from T by merging a new terminal to edge l, and denote by $(\bar{D}_l^+)$ the problem (D_l^+) where $x_i = \bar{x}_i$, $i \in \bar{E}_l$ and $l \notin \bar{E}_l \subset \{1, \ldots, n\}$. Then $z^{**} \geq \min_{l=1,\ldots,n} z^*(\bar{D}_l^+)$.*

3. The branch-and-bound algorithm

In our B&B implementation we employ an extension of the enumeration scheme introduced by Smith [19]. The B&B tree is initialized with a root node that corresponds to the unique full Steiner topology for the first three terminals. At any given node of the B&B tree at depth less than $N - 3$ there is a fixed full Steiner topology corresponding to a partial Steiner tree that uses a subset of the N original terminals. If the node is not fathomed, we select a new terminal to add to the partial tree and generate one child problem for each of its edges using the merge operation described in the previous section. We consider two strategies to select the next terminal node to be added to a partial Steiner tree at a given node in the B&B tree.

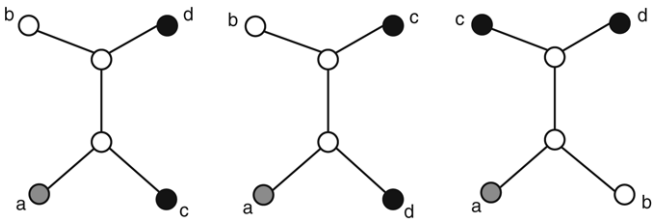

Fig. 2. Possible outcomes of two merge operations.

Smith : Select the first terminal from the input data file that is not in the current partial tree.
Smith+ : For each terminal that is not in the current partial tree compute the lower bounds $z^*(\bar{D}_l^+)$ described in Corollary 2. Select the terminal for which the largest number of children can be eliminated (via $z^*(\bar{D}_l^+) \geq v$, where v is the length of a known Steiner tree on all N terminals), breaking ties so as to maximize the sum of the bounds for the child nodes created.

The strategy used by the Smith+ algorithm is analogous to the *strong branching* methodology that has frequently been employed in B&B algorithms for integer or mixed-integer optimization [16]. The goal of this modification of Smith's original scheme is to minimize the size of the search tree by varying the order in which the terminal nodes are added. It is important to note, however, that Smith's argument for the correctness of his enumeration scheme implicitly assumes that the terminal nodes are added in a fixed order. In the next theorem we show that in fact Smith's enumeration scheme remains correct when the order in which the terminal nodes are added is varied.

Theorem 3. *Let T^{++} be a full Steiner tree on N terminal nodes that can be obtained from a partial Steiner tree T using a sequence of merge operations. Then T^{++} can be obtained using a sequence of merge operations that add the terminal nodes not present in T in any order.*

Proof. Assume for the moment that T^{++} is obtained from T using a sequence of exactly two merge operations, the first of which adds a terminal node c and the second a terminal node d. If c and d are merged with two arcs l and k that are both in T then it is obvious that the order of the merge operations can be reversed, resulting in exactly the same T^{++}. Alternatively assume that when d is added the arc that is used for the merge is one of the new arcs that was added to T to obtain the augmented tree T^+, as in Fig. 1. It is then easy to see that there are exactly three possible cases for how T^{++} differs from T. The three possibilities are shown in Fig. 2, where a and b denote the endpoints of the original arc l of T, as in Fig. 1. However interchanging c and d results in exactly the same three topologies, implying that whichever of these topologies corresponds to T^{++}, there is a sequence of merge operations that uses d followed by c resulting in the same T^{++}.

The above argument proves the theorem if T^{++} is obtained from T using two merges. If T^{++} is obtained from T using a longer sequence of merges, the same argument shows that any two terminal nodes that are adjacent in the merge sequence can be interchanged without changing T^{++}. Using a series of such interchanges we can transform any initial ordering of the terminal nodes to be added to any other ordering. □

In order to implement the Smith+ strategy we must choose a set of edges $E_l^+ \subset \{1, \ldots, n+2\}$ for the problem $(\bar{D}_l^+)$ used to estimate the effect of merging a new terminal node to edge l. It is obvious that adding edges to E_l^+ will tend to improve the bound obtained while increasing the cost of computing it. Our goal is to keep the computation of $z^*(\bar{D}_l^+)$ relatively inexpensive while hopefully obtaining some improvement over the current bound $z^*(D)$. The procedure that we use to define E_l^+ is to start with all edges that are incident to the new Steiner point and to the terminals (if any) that are adjacent to its neighbors. We then remove any edge k that is the unique edge adjacent to a Steiner node, since (as described below problem $(\bar{P})$ in the previous section) the values of the corresponding dual variables x_k are forced by the equality constraints of $(\bar{D}_l)$. In Fig. 3 we illustrate all possible resulting topologies for the subtrees of T^+ that contain only the edges in E_l^+. Note that $|E_l^+| \leq 7$ regardless of the size of the original tree T. It is also worth noting that in cases (g) and (i) the "subtree" of T^+ used to form $(\bar{D}_l^+)$ is actually the entire tree T^+; i.e. $\bar{E} = \emptyset$. These cases can only occur on levels 0 and 1 of the B&B tree, respectively, so at all deeper levels of the tree it is always the case that $|E_l^+| \leq 6$. It is also obvious that in case (a) the only feasible solution to $\bar{D}_l^+$ has $x_{n+1} = 0$, and no improvement in the bound will be possible; $z^*(\bar{D}_l^+) = z^*(D)$. This case corresponds to the edge

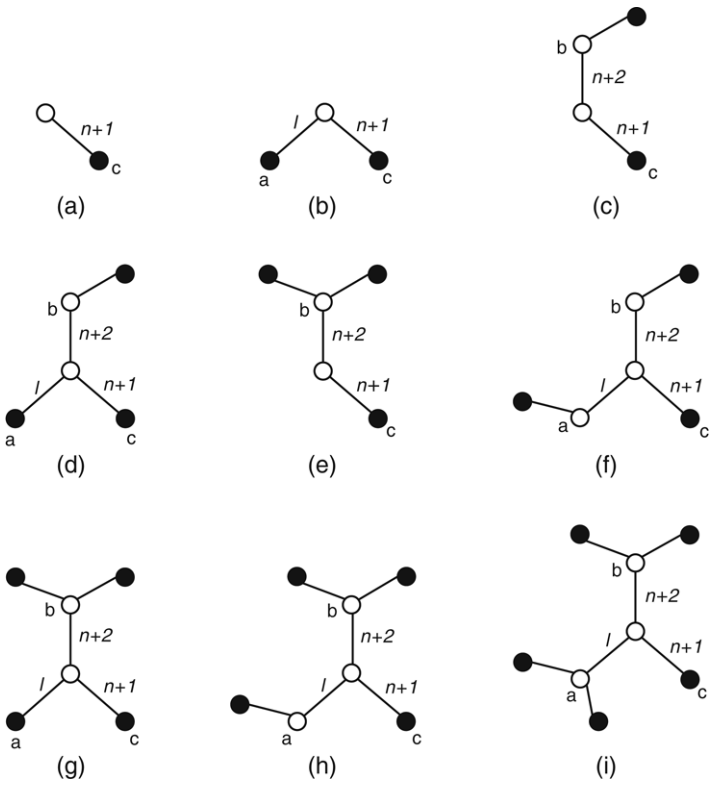

Fig. 3. Possible subtrees in subproblem $\bar{D}^+$.

l being adjacent to two Steiner nodes, both of which are adjacent to three Steiner nodes. In the following theorem we show that under a simple regularity condition a strict increase in the bound always occurs when one of the nodes adjacent to l is a terminal node, as in cases (b), (d) and (g) of Fig. 3.

Theorem 4. *Assume that edge l is adjacent to a terminal node c_l, and $s_l = \bar{s}_l \neq 0$ in an optimal solution of (P). Assume further that $c_l - c_{n+1}$ is not a positive multiple of $\bar{s}_l$, where c_{n+1} is the new terminal node being merged to edge l. Then $z^*(D) < z^*(\bar{D}_l^+)$.*

Proof. Consider case (b) in Fig. 3. Then $(\bar{D}_l^+)$ can be written

$$\begin{array}{ll} \max & c_l^{\mathrm{T}} x_l + c_{n+1}^{\mathrm{T}} x_{n+1} \\ \text{s.t.} & x_l + x_{n+1} = \bar{x}_l \\ & \|x_l\| \leq 1, \ \|x_{n+1}\| \leq 1. \end{array}$$

Furthermore it follows from the proof of Theorem 1 that $z^*(P) = z^*(\bar{P}^+) = z^*(\bar{D}^+)$ if and only if $x_l = \bar{x}_l$, $x_{n+1} = 0$ is optimal in $(\bar{D}^+)$, which holds if and only if $x_l = \bar{x}_l$ is optimal in the problem

$$\begin{array}{ll} \max & (c_l - c_{n+1})^{\mathrm{T}} x_l \\ \text{s.t.} & \|x_l\| \leq 1. \end{array}$$

Since $\bar{s}_l \neq 0$ we must have $\bar{x}_l = \bar{s}_l / \|\bar{s}_l\|$ [2], and for $x_l = \bar{x}_l$ to be optimal in the above problem it must be the case that $c_l - c_{n+1}$ is a positive multiple of $\bar{s}_l$ (implying that the new terminal node is in the affine hull of edge l). This completes the proof for case (b). The same argument also suffices if $\bar{D}_l^+$ contains more edges than in case (b) since fixing additional variables $x_i = \bar{x}_i$ cannot increase $z^*(\bar{D}_l^+)$. □

In the remaining cases of Fig. 3 edge l is adjacent to two Steiner nodes, at least one of which is adjacent to a terminal node. In this situation we can prove that $z^*(D) < z^*(\bar{D}_l^+)$ under more complex regularity conditions.

Theorem 5. *Suppose that edge l is adjacent to two Steiner nodes, one of which is adjacent to an edge k adjacent to a terminal node c_k. Assume that $s_l = \bar{s}_l \neq 0$ and $s_k = \bar{s}_k \neq 0$ in an optimal solution of (P), where $\bar{s}_l$ is not a multiple of $\bar{s}_k$. Let c_{n+1} be the new terminal node being merged to edge l, and assume that $c_{n+1} - c_k$ is not in the cone generated by $\bar{s}_l$ and $\bar{s}_k$. Then $z^*(D) < z^*(\bar{D}_l^+)$.*

Proof. Consider case (c) in Fig. 3, where k is the index of the second edge incident to node b. Then $(\bar{D}_l^+)$ can be written

$$\begin{array}{ll} \max & c_k^{\mathrm{T}} x_k + c_{n+1}^{\mathrm{T}} x_{n+1} \\ \text{s.t.} & x_k - x_{n+2} = \bar{x}_k + \bar{x}_l \\ & x_{n+1} + x_{n+2} = -\bar{x}_l \\ & \|x_k\| \leq 1, \|x_{n+1}\| \leq 1, \|x_{n+2}\| \leq 1. \end{array}$$

Furthermore it follows from the proof of Theorem 1 that $z^*(D) = z^*(\bar{D}_l^+)$ if and only if $x_k = \bar{x}_k$, $x_{n+1} = 0$, $x_{n+2} = -\bar{x}_l$ is optimal in $(\bar{D}_l^+)$, which holds if and only if $x_k = \bar{x}_k$ is optimal in the problem

$$\begin{array}{ll} \max & (c_k - c_{n+1})^{\mathrm{T}} x_k \\ \text{s.t.} & \|x_k\| \leq 1, \|x_k - \bar{x}_k - \bar{x}_l\| \leq 1. \end{array}$$

Since $\bar{s}_k \neq 0$ and $\bar{s}_l \neq 0$ we must have $\bar{x}_k = \bar{s}_k/\|\bar{s}_k\|$, $\bar{x}_l = \bar{s}_l/\|\bar{s}_l\|$ [2], and $\bar{x}_k \neq \bar{x}_l$ by the assumption that $\bar{s}_k$ is not a multiple of $\bar{s}_l$. It follows that $\|(\bar{x}_k + \bar{x}_l)/2\| < 1$, so the above problem satisfies a Slater condition and the KKT conditions are necessary and sufficient for optimality. Then $x_k = \bar{x}_k$ is an optimal solution if and only if there are multipliers $\lambda_k \geq 0$, $\lambda_l \geq 0$ such that

$$c_k - c_{n+1} + \lambda_k \bar{x}_k + \lambda_l \bar{x}_l = 0,$$

which is impossible under the assumptions of the theorem. The same argument applies if $(\bar{D}_l^+)$ has more arcs than in case (c) by first fixing additional variables $x_i = \bar{x}_i$. □

We use a depth-first search strategy to traverse the B&B tree so as to minimize the size of the queue of open problems. When there is more than one active node on the deepest level of the B&B tree, we use a best-first strategy to choose one among them. In this case we choose either the node corresponding to the partial Steiner tree with the smallest length (for the original Smith algorithm) or the one corresponding to the smallest lower bound from Corollary 2 (for our enhanced Smith+ algorithm).

4. Numerical results

In this section we present computational results that compare the performance of the algorithm proposed by Smith [19] with our enhanced version. Both algorithms were implemented in C and all runs were conducted on a 1.8 GHz Pentium CPU running under Linux. The solver MOSEK [1] was used to compute the length of each SMT for a given topology, and also the lower bounds corresponding to the addition of a new terminal node described in the previous section. Since MOSEK does not allow for the use of "warm-start" information when solving an SOCP, all computations are performed on a "cold-start" basis. An algorithmic framework for SOCP that permitted the use of warm-start information (like the dual simplex algorithm for linear programming) would be desirable in the branch-and-bound context. A simplex algorithm for SOCP that might allow for efficient use of warm-start information is suggested in [14].

The first problems we consider are a subset of the 46 instances from Soukup and Chow [20], available from the OR-Library [4]. These are all planar instances ($d = 2$) with between 3 and 20 terminals, except for one problem with $N = 62$. The optimal solutions for all of these instances were previously obtained by Winter and Zachariasen [24].

We begin with a detailed comparison of the B&B trees produced by the original and enhanced versions of Smith's algorithms on a single problem. Table 1 compares the two algorithms on problem 6 of the OR-Library set. This problem has 12 terminal nodes, so the B&B tree has 9 levels below the root. In Table 1 we give the number of nodes and CPU time expended at each level of the tree, as well as the fraction of nodes fathomed (in the column labeled "Fathom"). In addition, for the Smith+ algorithm we report (in the column labeled "Elim") the fraction of children of unfathomed nodes that were eliminated using the bounds described in the previous section. On this problem the Smith+ algorithm begins fathoming nodes on level 3 of the B&B tree, compared to level 6 for the Smith algorithm. In

Table 1
Comparison of B&B algorithms on problem 6

Level	Smith algorithm			Smith+algorithm			
	Nodes	Fathom	CPU secs.	Nodes	Fathom	Elim	CPU secs.
0	1	0	0.0	1	0	0	1.2
1	3	0	0.1	3	0	0	5.2
2	15	0	0.6	15	0	0.248	31.4
3	105	0	4.4	79	0.241	0.507	145.6
4	945	0	39.5	266	0.474	0.637	404.8
5	10,395	0	437.3	559	0.408	0.664	822.9
6	135,135	0.927	5,700.0	1,444	0.587	0.797	1,283.0
7	147,645	0.973	6,210.0	1,813	0.665	0.811	1,009.0
8	67,439	0.992	2,838.0	1,949	0.778	0.863	461.3
9	10,640		443.4	1,127			48.6
Total	372,323		15,673.3	7,256			4,213.0

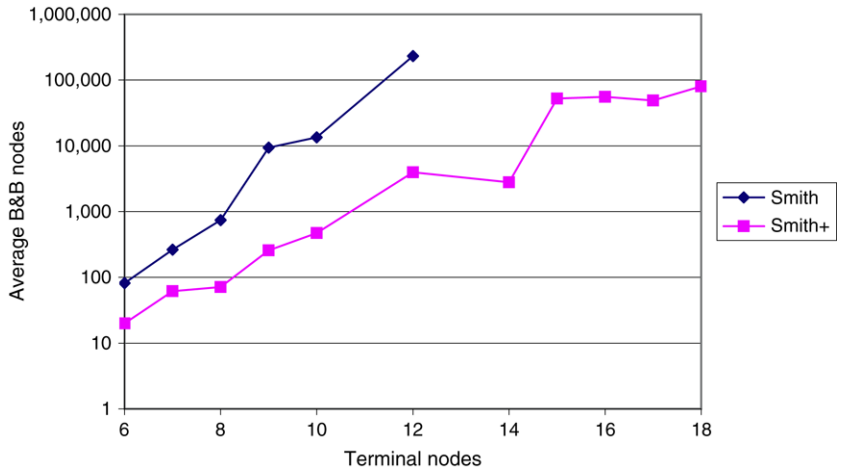

Fig. 4. Average number of nodes to solve OR-Library instances.

addition, the subtree bounds computed by the Smith+ algorithm are sufficient to eliminate a substantial fraction of the potential children of unfathomed nodes. Overall the subtree bounds and strong branching strategy used in the Smith+ algorithm reduce the number of nodes in the B&B tree by a factor of about 51 and the total CPU time required to solve the problem by a factor of about 3.7.

In Figs. 4 and 5 we summarize our results on 25 problems from the OR-Library instances. We solved all instances with $N \leq 14$ using the Smith+ algorithm, and all but one instance with $N \leq 12$ using the original Smith algorithm. In the figures we ignore problems with $N < 6$. For $6 \leq N \leq 12$ we give average results for the problems with each N that were solved by both the Smith and Smith+ algorithms. For larger N we give additional results for the Smith+ algorithm on the 5 problems with $N = 14$, and one problem for each N between 15 and 18. In Figs. 4 and 5 we report the average number of nodes in the B&B tree, and CPU time required, for problems of each size. Note the logarithmic scale used in both figures. On these problems the enhancements in the Smith+ algorithm reduce the average number of nodes in the B&B tree by a substantial factor that appears to be growing with N. For the smallest problems ($N \leq 8$) the time required to implement the strong branching strategy is not worthwhile, but for $N \geq 9$ the time required to solve the problems using Smith+ is lower than using the original Smith algorithm by a factor that again appears to be growing with N. Note that the root gap in the B&B tree is identical for the Smith and Smith+ algorithms, and in general will grow to very large values as N increases since the root lower bound corresponds to the minimal length of a Steiner tree that spans only 3 of the N terminal nodes.

In Table 2 we examine the complexity of the strong branching strategy used by the Smith+ algorithm on a single problem (problem 31 from the OR-Library). For each level of the B&B tree we give the number of times that each

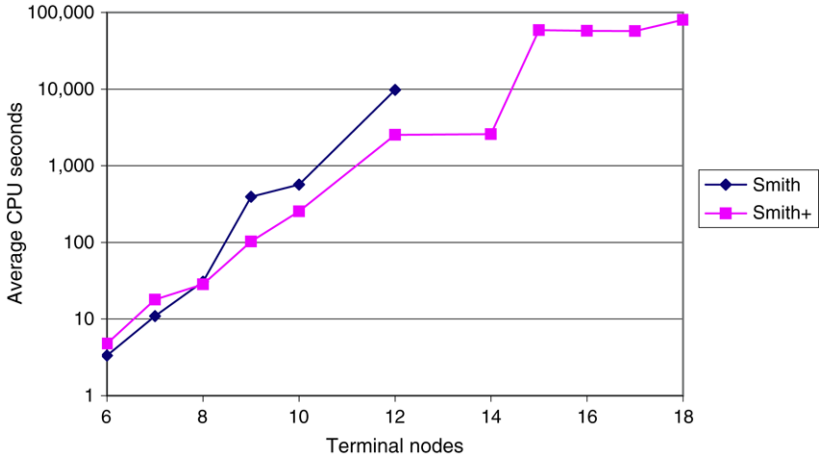

Fig. 5. Average CPU seconds to solve OR-Library instances.

Table 2
Branching structure for Smith+ on problem 31

Level	Next terminal node for branching										
	4	5	6	7	8	9	10	11	12	13	14
0			1								
1		3									
2	4			11							
3	57			22	1		1			1	
4	10				12	17	10	7	10	3	12
5	5				22	12	15	18	26	12	11
6	1				9	14	5	36	24	8	17
7					12	11	5	23	27	14	13
8					11	11	4	14	23	5	2
9					7	13	2	4	19	4	2
10					1	3	1		14	28	

Table 3
Performance on random instances with $N = 10$

d	Average B&B nodes			Average CPU seconds		
	Smith	Smith+	Factor	Smith	Smith+	Factor
2	16,821.6	105.0	160.2	717.8	68.4	10.5
3	55,222.2	1,652.4	33.4	2,334.1	753.5	3.1
4	368,762.8	13,685.6	26.9	16,153.0	5,735.6	2.8
5	470,321.8	9,250.0	50.8	20,805.6	4,680.5	4.4

terminal node was used to create the children of an unfathomed node. The table shows that the strong branching strategy used by Smith+ is potentially much more complex than a simple re-ordering of the nodes.

The OR-Library problems permit a good comparison of the Smith and Smith+ algorithms for different N, but all of the instances have $d = 2$. To investigate the effect of d we created problems for $2 \leq d \leq 5$ by generating $N = 10$ terminals randomly distributed in the unit hypercube. For each d we generated and solved 5 such instances using both the Smith and Smith+ algorithms. In Table 3 we report the average number of B&B nodes and CPU seconds for these problems, and also the factor improvement obtained using Smith+. Results on these problems demonstrate increasing difficulty to solve the problems with fixed N as d increases, especially for Smith's original algorithm.

It is clear that on both the OR-Library instances and the random problems with $2 \leq d \leq 5$, the factor by which Smith+ reduces the size of the B&B tree is considerably greater than the factor by which the CPU time is reduced.

This is to be expected given the effort required to execute the strong branching strategy as currently implemented. It is reasonable to expect that this time could be substantially reduced, with little increase in the size of the B&B tree, by only computing the bounds described in Section 2 for a subset of the most promising terminal nodes as determined by a heuristic criterion.

5. Conclusion

We present a new lower bound on the length of a Steiner tree that is a descendant of a given partial tree using Smith's branching scheme. This bound permits some children to be eliminated without computing the minimal lengths of Steiner trees for their topologies and also suggests the use of a strong branching scheme that varies the order in which the terminal nodes are added in an effort to minimize the size of the B&B tree. We show that Smith's enumeration scheme remains valid when the order in which the terminal nodes are added is varied. Our implementation of the algorithm uses conic formulations to obtain rigorous lower bounds for both the length of a Steiner tree with a given topology and the subtree problem that estimates the effect of merging a new terminal with an edge in the current tree. Numerical results demonstrate substantial improvements in both the amount of enumeration and total time required compared to Smith's original algorithm.

Acknowledgement

First author's research was conducted while visiting the Department of Management Sciences, University of Iowa, supported by a research fellowship from CNPq-Brasília-Brazil.

References

[1] E.D. Andersen, K.D. Andersen, The MOSEK interior point optimizer for linear programming: An implementation of the homogeneous algorithm, in: H. Frenk, C. Roos, T. Terlaky, S. Zhang (Eds.), High Performance Optimization, in: Applied Optimization 33, Kluwer Academic Publishers, 1999. See also http://www.mosek.com.
[2] K.D. Andersen, E. Christiansen, A.R. Conn, M.L. Overton, An efficient primal-dual interior-point method for minimizing a sum of Euclidean norms, SIAM J. Sci. Comput. 22 (1) (2000) 243–262.
[3] S. Arora, Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems, J. ACM 45 (5) (1998) 753–782.
[4] J.E. Beasley, OR-Library: Distributing test problems by electronic mail, J. Oper. Res. Soc. 41 (1990) 1069–1072. See also http://people.brunel.ac.uk/˜mastjjb/jeb/info.html.
[5] E.J. Cockayne, D.E. Hewgill, Exact computation of Steiner minimal trees in the plane, Inform. Process. Lett. 22 (1986) 151–156.
[6] E.J. Cockayne, D.E. Hewgill, Improved computation of plane Steiner minimal trees, Algorithmica 7 (2–3) (1992) 219–229.
[7] D.-Z. Du, W.D. Smith, Disproofs of generalized Gilbert–Pollak conjecture on the Steiner ratio in three or more dimensions, J. Combin. Theory A 74 (1) (1996) 115–130.
[8] M. Fampa, N. Maculan, Using a conic formulation for finding Steiner minimal trees, Numer. Algorithms 35 (2004) 315–330.
[9] M.R. Garey, R.L. Graham, D.S. Johnson, The complexity of computing Steiner minimal trees, SIAM J. Appl. Math. 32 (4) (1977) 835–859.
[10] E.N. Gilbert, H.O. Pollack, Steiner minimal trees, SIAM J. Appl. Math. 16 (1968) 1–29.
[11] F.K. Hwang, A linear time algorithm for full Steiner trees, Oper. Res. Lett. 4 (5) (1986) 235–237.
[12] F.K. Hwang, D.S. Richards, W. Winter, The Steiner tree problem, Ann. Discrete Math. 53 (1992), Elsevier, Amsterdam.
[13] F.K. Hwang, J.F. Weng, Hexagonal coordinate systems and Steiner minimal trees, Discrete Math. 62 (1986) 49–57.
[14] K. Krishnan, G. Pataki, N. Gupta, T. Terlaky, Towards a practical simplex method for second order cone programming, Presentation at INFORMS Annual Meeting, Denver, 2004.
[15] M. Lobo, L. Vandenberghe, S. Boyd, H. Lebret, Applications of second-order cone programming, Linear Algebra Appl. 284 (1998) 193–228.
[16] J.T. Linderoth, M.W.P. Savelsbergh, A computational study of branch and bound search strategies for mixed integer programming, INFORMS J. Comput. 11 (1999) 173–187.
[17] N. Maculan, P. Michelon, A.E. Xavier, The Euclidean Steiner problem in $\mathbb{R}^n$: A mathematical programming formulation, Ann. Oper. Res. 96 (2000) 209–220.
[18] Z.A. Melzak, On the problem of Steiner, Canad. Math. Bull. 4 (2) (1961) 143–148.
[19] W.D. Smith, How to find Steiner minimal trees in Euclidean d-space, Algorithmica 7 (1992) 137–177.
[20] J. Soukup, W.F. Chow, Set of test problems for minimum length connection networks, ACM/SIGMAP Newslett. 15 (1973) 48–51.
[21] W.D. Smith, J.M. Smith, On the Steiner ratio in 3-space. Working paper, Department of IEOR, University of Massachusetts, 1994.
[22] D.M. Warme, P. Winter, M. Zachariasen, Exact algorithms for plane Steiner tree problems: A computational study, in: D.Z. Du, J.M. Smith, J.H. Rubinstein (Eds.), Advances in Steiner Trees, Kluwer Academic Publishers, 2000, pp. 81–116.
[23] P. Winter, An algorithm for the Steiner problem in the Euclidean plane, Networks 15 (1985) 323–345.
[24] P. Winter, M. Zachariasen, Euclidean Steiner minimum trees: An improved exact algorithm, Networks 30 (1997) 149–166.